UNCLASSIFIED

AD 297 757

Reproduced
by the

ARMED SERVICES TECHNICAL INFORMATION AGENCY
ARLINGTON HALL STATION
ARLINGTON 12, VIRGINIA

UNCLASSIFIED

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

297757
CATALOGED BY ASTIA
AS AD NO.

MEMORANDUM
RM-3402-PR
FEBRUARY 1963

EXPERIMENTS IN LINEAR PROGRAMMING: Notes on Linear Programming and Extensions–Part 63

Leola Cutler and Philip Wolfe

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-3402-PR
FEBRUARY 1963

EXPERIMENTS IN LINEAR PROGRAMMING: Notes on Linear Programming and Extensions–Part 63

Leola Cutler and Philip Wolfe

This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force. Permission to quote from or reproduce portions of this Memorandum must be obtained from The RAND Corporation.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

PREFACE

This Memorandum presents the results of computational experience with certain variations on the "simplex method" of solving linear programming problems. The purpose of this study is to establish a basis for comparison of the efficiencies of various procedures in computation. Research on linear programming is conducted as part of The RAND Corporation's basic studies in mathematics.

This Memorandum should be of particular interest to those concerned with linear programming and of general interest to other mathematicians and computer specialists.

SUMMARY

This Memorandum summarizes the main results to date of the SCEMP Project (Standardized Computational Experiments in Mathematical Programming), involving the solution of nine linear programming problems using 30 variations of the simplex method. The statistics collected allow comparison of most of the variations which have been proposed in recent years and indicate the important features in the efficiency of linear programming routines.

ACKNOWLEDGMENT

We are indebted to Marvin Shapiro and Richard Clasen of RAND who did a substantial part of the computer programming. A number of members of the SHARE Linear Programming Project, notably David M. Smith of C-E-I-R, Inc., and L. Wheaton Smith of IBM, offered valuable advice. Much of the computing labor was defrayed through generous donations of machine time by C-E-I-R, Inc., Esso Research and Engineering Company, Phillips Petroleum Company, Shell Oil Company, Socony-Mobil Oil Company, and Standard Oil Company of California.

CONTENTS

PREFACE iii

SUMMARY v

ACKNOWLEDGMENTS vii

Section

I. INTRODUCTION 1

II. THE PROBLEMS 6

III. TERMINOLOGY 8

IV. STARTING BASES 12

V. THE FEASIBLE SOLUTION 17

VI. THE OPTIMAL SOLUTION 21

VII. SUBOPTIMIZATION 28

VIII. OPERATIONS AND FORMS 31

IX. ALGORITHMS COMPARED BY OPERATIONS 36

X. CONCLUSION 39

Appendix

THE SCEMP RUNS AND DATA 43

REFERENCES 45

LIST OF RAND NOTES ON LINEAR PROGRAMMING AND EXTENSIONS 47

I. INTRODUCTION

There are many ways to solve linear programming problems. The earliest of these, Dantzig's "simplex method,"[1] is the most widely used and no equally effective alternative is available. Many variations of the original simplex method have been proposed in the last few years. Computational experience seems to us the only way to properly compare the computational efficiencies of the variations; their behavior depends so strongly on features of the process which cannot be known in advance that a priori estimates of their effectiveness inspire little confidence. The purpose of the work reported here has been to compare some of the outstanding variations with each other in their work on actual linear programming problems, and to set some bench marks against which other procedures may be measured.

Under the title of "SCEMP"--Standardized Computational Experiments in Mathematical Programming--this work originated in 1960 at a meeting of the Linear Programming Committee of the SHARE organization (IBM 704-709-7090 users group), when it was suggested that some of the linear programming routines then forthcoming might serve the task of evaluating the procedures that had been discussed. The Committee maintains a file of problems from which those used here were selected; they are described in detail in the next section. A set of

statistic-collecting routines, modeled on an all-in-core, FORTRAN-coded linear programming routine for the IBM 704 and 7090[2] was coded and served as the basis for the computer routines used in the present tests. (The routines and the output of the tests have been retained and can be made available, but the routines are not recommended for general purposes.)

The nature of the output of these routines has been given in detail elsewhere.[3] Briefly, it consists in the following quantities for each simplex method iteration: the amount of infeasibility, the current value of the objective, the pivot row and column, the determinant of the basis, the number of product-form transformation entries, the number of arithmetic operations performed in each of several major subdivisions of an iteration, and the number of non-zero elements in certain arrays of interest.* At the end of a problem, the complete solutions are given as well as the "errors"--the extent to which the final solution fails of being both primal and dual feasible. All solutions obtained have been checked with those obtained by other routines on the same problems,[4] and the statistic-collecting features have been checked in detail for most of the runs by hand calculations of a small problem.

---

*See Sections III and VIII for definitions of terms used here.

The experimental data are organized by "runs," each of which consists of the entire set of test problems, by means of a routine embodying a particular algorithm variation. Of the runs done so far in the SCEMP project, 30 furnish the data used in this report; the others bear on matters not discussed here. Two kinds of data pertaining to a run have been used in this report: we consider the number of simplex method iterations, or changes of basis, required to reach a certain end--either the first feasible solution or the optimal solution of the problem (see Sections IV-VII); and we discuss the total number of arithmetic operations required (see Sections VIII and IX). The Appendix lists the raw data from which the figures presented in the sequel have been calculated.

Since the point of most of these experiments has been to compare alternative methods, the following general format has been used for the results. The appropriate data (e.g., number of iterations) for a particular run are chosen as a base. In order to compare another run with the base, the datum obtained in the comparison run for each of the test problems is divided by the corresponding datum for the base run; the resulting ratio is the proportion in which the measure has been reduced by use of the compared procedure. For example, suppose that Algorithm I took 20 iterations to solve

problem 1D and 30 to solve problem 2A; and that Algorithm II took 14 and 24 iterations, respectively. Choosing Algorithm I as the base, the comparative results would be given as in Table 1-1.

Table 1-1

Algorithm II Compared with Algorithm I

Problem	1D	2A	avg.	c.v.
Alg. II	.70	.80	.75	.07

Note that usually the average of the ratios is given, as well as their coefficient of variation (the standard deviation divided by the average). The problems will be listed in order of their number of constraints. The ratios all have equal weights in the averaging, but the average could be viewed as an average of the data of the compared run weighted by the reciprocals of the corresponding data of the base run. For this reason, the average is a somewhat fairer measure when the data of the base are larger than those of the compared run. Owing to the arithmetic of averaging, if II were chosen as the base and I as the compared run, the resulting average would be greater than the reciprocal of that of 1-1.

Some gaps appear in the tables that follow. The largest problem cannot be run on the routines using the standard form of the simplex method, and two other problems were omitted from some "feasible solution" runs

because they had starting feasible solutions.

A good deal of special terminology is used in describing the computations. Special terms are usually defined by context at their first appearance, which is signaled in underlining. Most of them are introduced in Sections III and VIII. While the terms "method" and "procedure" are used interchangeably in a very general way, we use the term algorithm to refer to any particular version of the simplex method which chooses the pivot column and the pivot row in a particular manner, regardless of the way in which the data used in making the choice are obtained. Thus Sections III to VII study only algorithms and the data--principally operation counts--associated with them, while the remaining sections study, in part, different methods of performing the same algorithm.

II. THE PROBLEMS

The linear programming problems on which our experiments were conducted were drawn from the file of thirteen problems maintained by the Test Problems and Experiments Committee of the SHARE Linear Programming Project. The problems, submitted by various members of the Committee in 1959 and 1960, were all used as production problems in their businesses; the majority arose in oil refining studies. None were especially constructed for test purposes, or thought "pathological." The original problems are available through the Committee.

Four of the problems were not used here. Problem 1C is too small, 4A is too large, and 3A and 3B had awkward input features. Thus our work was done with the nine problems of Table 2-1.

Throughout this report the problems are listed in the order of their numbers of constraints. In Table 2-1, the "name" identifies a problem in the Committee's files. All the problems are formulated as problems of minimizing a linear objective function under linear equality constraints; some problems have several alternative objective functions, so that there are always one or more "additional rows." In our runs the highest-numbered objective row was used and the remainder ignored.

The "number of variables" includes all the variables of the problem but no "artificial" variables. The

"number of entries" is the number of non-zero quantities appearing among the constraints and objectives. Some further data regarding starting bases for the problems are given in Sec. IV.

Table 2-1

The Test Problems

Name	1D	2A	1E	1A	5A	1G	1F	2B	1B
Number of constraints: M	27	30	31	33	34	48	66	96	117
Number of additional rows: K	1	2	8	1	1	1	1	3	1
Number of variables	45	103	106	64	78	102	135	162	253
Number of entries	252	811	855	245	391	462	644	897	1210

III. TERMINOLOGY

In order to describe the algorithms studied, we develop here some of the terminology connected with the simplex method. It is not intended to discuss the procedure itself, which is done in many standard works.[5,6] The discussion in this section is entirely in terms of the standard form of the simplex method; the other forms are dealt with in Sec. VIII.

Let a linear programming problem have N variables $x_1,..,x_N$ and M equation constraints. At any stage in the simplex method solution there is defined a *basis*, which is a set of M *basic variables*, say $x_{j_1},..,x_{j_M}$; let the remaining variables be $x_{j_{M+1}},..,x_{j_N}$. The current *tableau* is the set of coefficients of the linear equations

$$x_{j_1} + a_{11}\, x_{j_{M+1}} + \ldots + a_{1,N-M}\, x_{j_N} = b_1 ,$$

$$\vdots$$

$$x_{j_M} + a_{M1}\, x_{j_{M+1}} \quad \ldots \quad a_{M,N-M}\, x_{j_N} = b_M ,$$

which are uniquely defined by the current basis and the requirement that this set be equivalent to the linear equations defining the original problem. We say that the basic variable x_{j_i} *occupies position i* in the basis, for $i = 1,..,M$.

It is further supposed that the objective function to be minimized is expressed at this time in terms of the nonbasic variables as

$$c_1 x_{j_{M+1}} + \ldots + c_{N-M} x_{j_N} + z_o ;$$

the coefficients c_j are the reduced costs. (The quantities a_{ij} and c_j defined here commonly carry a superior bar to indicate that they change in each iteration; we omit the bar.) The basic solution of the equations above is obtained by setting all nonbasic variables to zero, giving the basic variables the values $x_{j_1} = b_1$, etc., and the objective the value z_o.

In a single iteration of the simplex method a pivot column J (where j_{M+J} is the index of a nonbasic variable) and a pivot row I of the tableau are chosen, and the roles of the basic variable occupying position I and the nonbasic variable associated with column J are interchanged, new data of the form of the equations above being obtained by pivoting on the entry a_{IJ} of the tableau. The value of the objective changes by the amount $c_J b_I / a_{IJ}$.

Some M variables must be chosen as the starting basis for the procedure. If not all these variables belong to those of the original problem, the remainder are artificial. Each instance of a basic variable whose current value is negative, or of an artificial variable

whose value is not zero, is an infeasibility. A basic solution having no infeasibilities is feasible. When it is necessary to adjoin artificial variables in order to have a starting basis, we always adjoin for each a column of coefficients of the form [0,..,1,..,0] to the original problem, the single "1" of the column lying in a row corresponding to an otherwise unoccupied position of the basis. When there are infeasibilities, a separate objective function involving them is defined, and it is required that this infeasibility objective be minimized. The process of minimizing that objective is Phase One; the subsequent minimization of the proper objective, once a feasible solution is obtained, is Phase Two.

In the sequel we refer to the ordinary simplex method, by which we mean the simplex method as most commonly presented, except that we extend the usual procedure for the choice of pivot row to that of the "composite algorithm."(7)

In Phase Two the procedure is quite ordinary. A pivot column J is chosen so that c_J is minimal (if all are non-negative, the current solution is optimal). Then the pivot row I is chosen so that after pivoting the current solution will still be non-negative: I is the i which minimizes b_i / a_{iJ} for all $a_{iJ} > 0$. If $b_I = 0$--degeneracy--should happen, then I is chosen as the i maximizing a_{iJ} among all i for which $b_i = 0$. (This rule is not known to prevent "cycling," but is very effective in practice.)(8)

In Phase One the objective is defined as the sum of the infeasibilities: $\Sigma \{x_j : x_j < 0$ or x_j artificial$\}$. The reduced cost for the nonbasic variable j is then

$$\Sigma \{a_{ij} : b_i < 0\} - \Sigma \{a_{ij} : b_i > 0 \text{ and position } i \text{ artificial}\} .$$

The pivot column J is chosen for minimal reduced cost, and the pivot row I so that no variable non-negative in the current solution becomes negative after pivoting: I is the i which achieves the smaller of the two ratios $\text{Min}_i \{b_i/a_{iJ} : b_i , a_{iJ} > 0\}$, $\text{Max}_i \{b_i/a_{iJ} : b_i , a_{iJ} < 0\}$. A somewhat more complicated rule is needed for degeneracy. In the absence of negative b_i , the pivot-row rule operates just as in Phase Two.

IV. STARTING BASES

The basis with which a problem is started naturally has a great influence on the number of iterations required to solve it. In practice one often attempts to guess a starting basis which will be as nearly feasible and optimal as possible; a sophisticated routine will make good use of such a guess even if the basis is incomplete, infeasible, or singular. The three methods studied here do not, of course, make any use of special information about the problem; they assume complete ignorance, and may be used with any problem.

N basis: When no starting basis is specified, a full set of M artificial variables is adjoined to the problem and constitutes the starting basis.

S basis: By singleton we mean a variable having only one non-zero entry, and that positive, in the equations of the initial tableau. An S basis is a starting basis consisting of a maximal set of singletons, with artificial variables used as necessary for the unfilled positions. (The computational cost of pivoting on singletons is almost nothing, and feasibility is improved if all the original right-hand sides are non-negative.)

F basis: A full basis was produced by this procedure: first, an S basis was chosen; subsequently, each column of the tableau was examined, and pivoted into the basis if it had a non-zero entry corresponding to any unfilled basis

position. The only basis positions left unfilled by this procedure are those corresponding to redundant constraints. Naturally, the resulting basis is not likely to be primal or dual feasible. Other procedures for obtaining a full basis have been tried but not yet fully evaluated; they do not seem to offer much advantage over the above.

Table 4-1 describes the bases resulting from the use of procedures S and F. All the data are proportions, the number of variables in a given category being divided by the number of constraints in the problem. The last two lines constitute the proportion of infeasibilities in the starting basis. Note, however, that artificial variables initially at zero level tend to become non-zero before they are eliminated, so that for an S basis the total number of artificials is the better measure of infeasibility.

Table 4-1

Starting Basis Characteristics

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B
					(S basis)				
Singletons used	.19	.87	.19	.30	1.00	.90	.65	.86	.29
Positive artificials	.81	.00	.19	.70	.00	.10	.35	.05	.60
Zero artificials	.00	.13	.61	.00	.00	.00	.00	.08	.11
					(F basis)				
Negative variables	.41	.00	.42	.30	.00	.23	.26	.10	.46

Note that the proportion of infeasibilities in the F basis runs a little more than half the proportion in the S basis. We view this as accounting for the advantage, to be seen below, of the F basis over the S basis.

We are mainly interested in the number of simplex method iterations required to obtain the first feasible solution after the starting basis has been constructed. (In Sec. IX the effect of the work required to produce the starting basis, as included in the total work to solve the problem, is considered.) Bases N and S have been used with two algorithms: the ordinary procedure and the "ratio pricing" procedure, described in Sec. VI. The results are summarized in Table 4-2. The first line compares basis S (run 21) with basis N (run 5, used as the base), for the ordinary algorithm; the second line compares basis S (run 6) with basis N (run 8, used as base) for the ratio pricing algorithm. The ratios thus represent the proportion in which the number of iterations in Phase One is decreased by using an S basis rather than an N basis.

Table 4-2

S Basis Compared to N Basis for Two Algorithms

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B	Avg.
Ordinary algorithm	.89	.00	.69	.61	.00		.51	.42	.83	.50
Ratio pricing	.86	.01	.85	.82	.00	.40	.58	.83		.54

Evidently use of an S basis entails, on the average, a saving of about 48% in the number of iterations required for Phase One.

Comparison of bases S and F has been made in each of three algorithms, with the number of iterations for basis S taken as the base data: the ordinary algorithm (runs 21 and 39, respectively); the sequential procedure (runs 31 and 36); and the least-infeasibility procedure (runs 33 and 37). The last two procedures are discussed in Sec. VI. Table 4-3 summarizes these, omitting problems 2A and 5A because their starting S and F bases are feasible.

Table 4-3

F Basis Compared to S Basis

Problem	1D	1E	1A	1G	1F	2B	1B	Avg.
Ordinary algorithm	.32	.71	.48	1.62	.05	.69	.53	.63
Sequential	.37	1.00	.37	2.43	.11	1.07	.55	.84
Least-infeasibility	.33	.94	.35	2.27	.09	.93	.78	.81

The overall average of these proportions is 0.76, predicting a saving of 24% in use of an F basis rather than an S basis.

We conclude that in the absence of other knowledge of the problem, an F basis should be used. Some linear programming routines[9] make it possible to use a mixed procedure, entering a known partial basis and subsequently completing it in an arbitrary manner.

We may try to predict the number of iterations Phase One requires using the ordinary algorithm. Each entry in Table 4-4 is obtained by averaging, for all problems, the number of iterations taken using the basis N, S, or F divided by one of three possible measures of problem difficulty--the number, M, of constraints, the number of non-singletons, or the number of infeasibilities in the F basis. Thus, for example, the number of iterations required using an S basis is expected to be .78 M. The coefficients of variation are given in parentheses. It is disappointing that the number of constraints is a better basis for prediction than the more informative measures.

Table 4-4

Phase One Iterations vs. Measures of Problem Difficulty

		Measure					
		M		Number of non-singletons		Number of negatives in F basis	
Starting Basis	N	1.69	(.3)				
	S	.78	(.8)	2.13	(.9)		
	F	.56	(.6)	2.07	(1.1)	2.12	(.8)

V. THE FEASIBLE SOLUTION

In general, Phase One, the task of obtaining a first feasible solution, is accomplished by employing the simplex method to minimize some measure of the infeasibility of a solution. The five procedures studied here employ four different measures of infeasibility. In all of them the measure constitutes an objective function whose reduced costs are calculated so that the choice of pivot column can be made by the ordinary rule. In all but the "extended composite" algorithm the ordinary rule of pivot row selection is used.

The ordinary procedure is described in Sec. III.

The extended composite procedure[10] differs from the ordinary in choice of pivot row. After the pivot column has been chosen in the ordinary way, the pivot row is selected so that the sum of infeasibilities after pivoting will be minimized; variables are allowed to change sign freely. Thus I is defined by $\theta_o = b_I/a_{IJ}$, where θ_o minimizes $\Sigma_i\{|b_i - \theta\, a_{iJ}| : b_i - \theta\, a_{iJ}$ is infeasible$\}$.

In the sequential procedure, the infeasibility is corrected one component at a time, in order. At any iteration, let i_o be the least i for which some b_i is infeasible, and x_r the corresponding variable. The objective for minimization is defined as x_r if position i_o is artificial and b_{i_o} is positive, or as $-x_r$ if b_{i_o} is negative. (The reduced cost for column j will

then be just $a_{i_o j}$ or $-a_{i_o j}$.) During the procedure, row i_o will be made feasible, feasibility on the previous rows being preserved.

The least-infeasibility procedure is like the sequential, except that at each iteration the index i_o is taken so that x_r is minimal among all infeasible variables; the index may increase or decrease.

The fudge procedure, but not its name, is due to Gass.(5*) A problem having negative solution values is augmented by a single artificial variable and subjected to a transformation yielding non-negative solutions for the augmented problem. Specifically, the tableau is augmented by a column containing the entry -1 in each row having $b_i < 0$ and zeros elsewhere; and the desired tableau is obtained by pivoting in the I^{th} entry of the added column, where $b_I = \min_i b_i$. Subsequently the sum of all the artificial variables is minimized using the ordinary algorithm; when it has been reduced to zero, a feasible solution is at hand. (Of course, other means of getting feasible could be used once the negativity has been removed.)

Table 5-1 lists the runs done using these five procedures, indicated at the left. The starting basis used is listed at the top.

---

*Pp. 120-125.

Table 5-1

Feasible Solution Runs

	N	S	F
Ordinary algorithm	5	21	39
Extended composite algorithm	(5)	(21)	38
Sequential algorithm		31	36
Least-infeasibility algorithm		33	37
Fudge procedure	(5)	(21)	32

Runs indicated in parentheses were not done, since the same results would have been obtained as in the run whose number is given.

Tables 5-2 and 5-3 give the results for these procedures, for bases S and F, relative to the ordinary procedure. The last line of each table is the proportion of infeasibilities in the starting basis for each problem.

Table 5-2

Phase One, S Basis, Relative to Ordinary Algorithm

Problem	1D	1E	1A	1G	1F	2B	1B	Avg.
Sequential	.96	.94	1.52	1.00	.98	.97	1.04	1.06
Least-infeasibility	1.08	1.03	1.36	1.05	1.24	1.12	.94	1.12
Proportion infeasibility	.81	.19	.70	.10	.35	.05	.60	

The amount of infeasibility does not seem to affect the relative efficiencies of these methods much, although it does affect the total work done, as the data for runs 21, 31, and 37 in the Appendix, or those of Table 4-4, show.

Table 5-3

Phase One, F Basis, Relative to Ordinary Algorithm

Problem	1D	1E	1A	1G	1F	2B	1B	Avg.
Extended composite	1.00	1.08	1.42	.65	1.00	1.04	1.01	1.03
Sequential	1.12	1.33	1.17	1.50	2.00	1.51	1.08	1.39
Least-infeasibility	1.12	1.37	1.00	1.47	2.00	1.51	1.39	1.41
Fudge	1.25	1.08	1.25	1.00	2.00	1.34	1.01	1.27
Proportion infeasibility	.41	.42	.30	.23	.26	.10	.46	

The results pretty well establish the ordinary procedure as superior in getting feasible. Its objective is responsible, since the minimization algorithm is the same in all the runs. It seems that by moving in a direction tending to minimize the sum of all the infeasibilities we give more chance to a number of infeasibilities to leave, while the sequential and least-feasible procedures, concentrating on a single variable at a time, are too single-minded. Since several negative infeasibilities can be removed in one iteration, while only one artificial variable can, it is reasonable that the difference is more decisive for F bases than for S bases.

The extended composite procedure is somewhat disappointing. It might work better if, at the expense of considerably more calculation, the pivot column were chosen by the same criterion as is the pivot row.

VI. THE OPTIMAL SOLUTION

Of greatest interest to the ordinary user is the amount of work required to solve a complete problem. In this section six algorithms are compared in the number of iterations required to obtain an optimal solution. All but one of these are designed to handle artificial variables; for them, the ordinary Phase One objective--the sum of all infeasibilities--is used; this was found most efficient in Sec. V. Unless otherwise noted, each procedure uses the same method for minimizing its objective in Phase One as it does in Phase Two, only the definition of the objective changing between the phases. Similarly, each procedure (except the "symmetric") uses the ordinary choice of pivot row. They differ primarily in the manner of choosing the pivot column.

The ordinary procedure was described in Sec. III.

The positive-normalized procedures (PN1 and PN2) can be viewed as representative of those proposals which aim at eliminating the effects of bad scaling of the problem data by dividing the reduced costs, used in choosing the pivot column, by some combination of the coefficients a_{ij}. The first of the two considered here, proposed by Dickson and Frederick,[11] uses the formula $d_j = c_j^2 / (c_j^2 + \Sigma_i\, a_{ij}^{+\,2})$, where a_{ij}^+ is the "positive part" of a_{ij}, choosing the pivot column as that j for which d_j is maximal for $c_j < 0$. The procedure PN2 is essentially

this, using instead the formula $d_j = c_j^2 / \Sigma_i a_{ij}^{+2}$, which gives the same result.

The PN1 procedure employs the slightly simpler formula $d_j = c_j / \Sigma_i a_{ij}^+$, with the pivot column chosen for minimal d_j.

The greatest-change procedure was described long ago,(1) but has been little used. That column is chosen which, after pivoting, will give the greatest decrease in the value of the objective; it is the j which minimizes the expression $c_j \min_i\{b_i/a_{ij} : a_{ij} > 0\}$ for the change of the objective.

The ratio pricing procedure was suggested informally by Markowitz some time ago. It differs from the ordinary procedure only in Phase One. Letting w_j be the reduced cost for the infeasibility objective then, and c_j be the reduced cost for the proper objective, the pivot column j is chosen so as to maximize c_j/w_j for $w_j < 0$; we obtain the largest possible improvement in the proper objective per unit change of infeasibility. It may be viewed as an application of parametric linear programming:(5) defining ϕ^* at each iteration as the largest ϕ such that $c_j + \phi w_j \geq 0$ for all $w_j < 0$, the pivot column is chosen so as to increase ϕ^*. Evidently when ϕ^* becomes sufficiently large we have all $w_j \geq 0$, and Phase One is ended. It turns out that almost all c_j are then non-negative, too, so that Phase Two is quite short. The

aim of the procedure is to obtain a first feasible solution which is nearly optimal; the data of the Appendix for run 6 show that it does this well.

The symmetric procedure of Talacko[12] is employed only with a full basis; it may take either "primal" or "dual" simplex method steps. For one iteration: Among those columns with negative reduced costs and those rows whose basic variables are non-negative, a potential pivot is determined using the greatest-change procedure as described above; and among columns with positive reduced costs and rows with negative variables, a potential pivot is determined using the dual of the greatest-change procedure (for which the greatest increase of the objective is sought). That pivot is used for which the magnitude of the objective change is greater. If a step of the first kind is taken, all non-negative basic variables stay non-negative. The procedure does not always terminate in a solution of the problem;[12*] but it did for the test problems.

The parametric procedure of Dantzig[13] is likewise employed conveniently only with a full basis, and takes either primal or dual steps. It is generally like the symmetric procedure, but its termination is known. At the beginning of procedure a column of right-hand side

---

*P. 10.

changes is added to the tableau so that when the column is multiplied by an initial value ($\phi_0 > 0$) of the parameter ϕ and added to the current basic solution the result will be non-negative; and likewise a row of cost changes is added to the tableau so that when multiplied by the initial value of ϕ and added to the current reduced costs, the result will be non-negative. In this manner a problem involving the single parameter ϕ has been defined which is both primal and dual feasible when $\phi = \phi_0$ and is equivalent to the original problem when $\phi = 0$. The procedure consists in then reducing ϕ using the pivot choice rules of the parametric linear programming algorithms,[5,13] which will give a sequence of primal and dual feasible solutions for a decreasing sequence of values of ϕ terminating in zero.

Table 6-1 compares all these, taking the ordinary procedure as the base (run 21).

Of the runs with singleton basis, the positive-normalized procedures are outstanding, and overall the greatest-change procedure with full basis is best. Unfortunately the positive-normalized procedures have not yet been tried with full bases; they might perform even better. Incidentally, the data of the Appendix show that, with the natural exception of ratio pricing, the differences among the procedures are reflected in Phase One in about the same way as in the entire process.

Table 6-1

Algorithms and Bases Compared with Ordinary, S Basis

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B	Avg.	c.v.
					Singleton basis						
PN1--run 10	.76	.80	.96	1.00	.73	.66	.84	.82		.82	.1
PN2--run 14	.83	.82	.98	.93	.76	.73	.66	.90		.83	.1
Greatest-change run 15	.70	1.10	1.23	.76	.86	.65	.73	1.21		.91	.2
Ratio pricing run 6	.59	1.62	1.42	.81	.95	.82	.55	1.35		1.01	.4
					Full basis						
Ordinary--run 39	.63	1.24	.92	.52	1.00	1.21	.45	.76	.76	.83	.3
Symmetric--run 40	.43	.82	1.53	.45	.84	.74	.49	.73		.75	.4
Greatest-change run 41	.39	.82	.68	.55	.84	.94	.38	.90	.27	.64	.4
Parametric run 59	.39	1.74	1.51	.57	1.03	1.86	.64	.96		1.09	.5

(Note: If problem 1B is eliminated from 39 and 41, the averages are .84, .69.)

The data of Table 6-1 allow the symmetric and greatest-change procedures to be compared directly with the ordinary procedure with full basis. Using run 39 as base, the averages and coefficients of variation obtained are: symmetric algorithm, .92, .3; greatest-change algorithm, .78, .3; parametric algorithm, 1.25, .2. The relative efficiencies of these procedures are not changed much by calculating them from the different base run.

It is of considerable interest to find some means of predicting the work needed for a problem about which little is known. In Table 4-4 it was found that the number M of constraints was the best guide of those studied to the number of iterations for Phase One; we shall use it also in connection with the total iterations required. Table 6-2 thus lists the number of iterations required to solve each of the problems using the ordinary algorithm divided by M . It would appear that rule of "2M iterations" from folklore is fairly good when a singleton basis is used.

Table 6-2

Iterations/Constraints for Ordinary S basis (run 21)

1D	2A	1E	1A	5A	1G	1F	2B	1B	avg.	c.v.
2.00	1.67	1.71	1.27	1.09	1.29	1.83	1.18	3.33	1.71	.4

The corresponding data for the algorithms of Table 6-1 can be found by multiplying the entries of 6-2 by those of 6-1. The averages thus obtained appear in Table 6-3.

Table 6-3

Summary of Iterations, ts

Algorithm	Run	Average	c.v.
	Singleton basis		
Ordinary	21	1.71	.4
PN1	10	1.24	.2
PN2	14	1.24	.2
Greatest-change	15	1.36	.3
Ratio pricing	6	1.50	.4
	Full basis		
Ordinary	39	1.39	.4
Symmetric	40	1.13	.5
Greatest-change	41	.98	.2
Parametric	59	1.60	.3

A more detailed examination of the data seems to show that the dependence of the number of iterations on M could be better expressed by a formula of the form $k M^b$, where b is slightly less than one, but this is not clear. Using a singleton basis an estimate of between M and $3M$ iterations will almost always be correct.

VII. SUBOPTIMIZATION

Versions of suboptimization have been used for some time in linear programming routines bothered by small core size, but the advantages of a version of it for routines for which core size is no particular handicap were first exploited by D. M. Smith.[14] As used here, the course of the solution of a problem consists of a number of passes, at the beginning of each of which some number L of nonbasic columns is selected as a set of candidates for pivoting (those having the L minimal reduced costs are chosen). During the pass no other nonbasic columns are considered; simplex method iterations are performed using the selected columns until the objective has been minimized on that subset. (A basic column which becomes nonbasic during the pass is not further considered.)

The number L of candidates is an important parameter; values of 2, 3, 5, and 8 were used here. During a pass, any of the various means of selecting a pivot column discussed previously might be used in minimizing the objective on the candidates. Three were tried here: the ordinary procedure of minimal reduced cost; the greatest-change procedure; and the procedure PN1.

Both the number of iterations and the number of passes required to solve a problem are of interest. In the table below, the numbers required are all compared with the number of iterations used by the ordinary simplex method

(run 21), which would be the number of passes for any of the algorithms for $L = 1$. Only the averages and coefficients of variation are given for these runs; the individual data fluctuate considerably less than in most of our experiments. An interesting feature of the raw data not reflected in the averages is that the greatest-change procedure commonly requires fewer iterations under suboptimization than does the ordinary procedure without it, which is generally not the case for the other methods.

Table 7-1

Suboptimization Runs Compared to Ordinary Algorithm

Run	Algorithm	L	Iterations		Passes	
			average	c.v.	average	c.v.
22	ordinary	2	1.15	.2	.72	.2
27	"	3	1.28	.2	.60	.2
28	"	5	1.26	.2	.45	.2
29	"	8	1.31	.3	.37	.4
23	greatest-	2	1.07	.2	.72	.2
24	change	3	1.08	.2	.59	.2
25	"	5	1.08	.3	.45	.3
26	"	8	1.13	.3	.40	.3
42	PN1	2	1.15	.1	.72	.2
43	"	3	1.22	.2	.58	.2

The term "pass" arises from the fact that it is only necessary to consult the data for the entire problem once during a pass; the data which have to be retained for the subsequent suboptimization are much fewer. This fact makes it particularly valuable in product form routines

and those which use tapes extensively. (The three main forms of the simplex method are discussed in Sec. VIII.) The significance of the statistics above depends on the form of routine used. In the product form, the total work done depends largely on the number of passes; in the standard form, on the number of iterations; and the explicit form is intermediate. Thus suboptimization is of value in the product form even for all-in-core routines, but not in the standard form. Three production linear programming routines now use it in the manner described. They are all product form routines, one using the ordinary algorithm with $L = 2$,[9] another the greatest-change algorithm with $L = 2$,[14] and the third has options for either algorithm and any L up to 5.[15]

VIII. OPERATIONS AND FORMS

So far we have been concerned only with the number of iterations required to solve a problem. A better guide to the computational efficiency of a procedure is the number of floating-point arithmetic operations performed--the work which must be done no matter how the algorithm is implemented. While logic and bookkeeping time are usually appreciable, and vary between different algorithms and different forms of the simplex method, it is precisely in such non-arithmetic work that computers and programming systems differ the most. Having programmed each of the procedures studied here as economically as we could from the standpoint of arithmetic, we feel that the results on arithmetic work come close to a machine-independent measure of efficiency.

Although it would be possible to count separately each elementary operation, it turns out that there are only three combinations of elementary floating-point operations used significantly often in each of the major subdivisions of an iteration: addition and multiplication; division and subtraction; and addition alone. Each of the following three groups is thus called one operation:

1 floating add and 1 floating multiply	(17.4)
1 floating divide and 1 floating add	(19.4)
3 floating adds	(19.2).

The average number of 7090 cycles taken by each combination is given in parentheses. While some error is made in considering all these equivalent, it is very small, because the first combination accounts for almost all the calculations. In all cases (except for a portion of the reverse-transformation calculation in the product form) an operation is counted only when both operands are non-zero.

There are many ways of calculating the data required for the steps of the simplex method. In all of them the data used in the ordinary procedure are obtained, but in different ways. The three main forms of the method are described below in outline; the details may be found in the literature.[5,6] In considering the number of operations performed in one iteration in any form, it is convenient to have a priori estimates in terms of M (the number of constraints), N (the current number of variables), and $M + K$ (the total number of rows of data). In the formulas below, factors of proportionality θ between zero and one reflect the fact that operations involving zero data are not counted; and quantities of order smaller than M^2 are disregarded.

The standard form is done just as the ordinary procedure is described in Sec. III. Pivoting in the tableau is most of the work (requiring $\theta_1(M+K)(N-M)$ operations).

Both forms of the "revised simplex method" calculate

needed items of the tableau by multiplying parts of the original matrix A by parts of the inverse, the inverse of the $M + K$-order matrix consisting of the basic columns of A. The reduced costs are obtained by multiplying A by the prices, that row of the inverse corresponding to the objective row of A ($\theta_2(M + K)(N-M)$ operations); the selected pivot column of the tableau is obtained by multiplying the appropriate column of A by the inverse (the number of operations required for this and the remaining steps differs for the two forms); the pivot row is selected as usual; and pivoting is done both in the inverse and the current solution.

In the explicit form, or the "revised simplex method with explicit form of the inverse," the inverse is a square $M + K$-order matrix, all of which is pivoted in at each iteration. Pivoting requires $\theta_4(M + K)^2$ operations, and the prior multiplication for the pivot column requires $\theta_3(M + K)^2$ operations.

In the product form, or the "revised simplex method with product form of the inverse," the inverse is maintained as a sequence of transformations, each of which, having at most $M + K$ nonzero entries, constitutes the nontrivial portion of the pivot column of the tableau as of some previous iteration. Applied appropriately, these transformations accomplish the work of matrix multiplication required by the revised simplex method. In a pivot step these data are not altered but are augmented

by one more transformation. Their total number is generally somewhat less than $(M + K)^2$, and most, but not all, of them are used once in obtaining the prices ($\theta_5(M + K)^2$ operations) and the pivot column ($\theta_6(M + K)^2$ operations). The number of accumulated transformations is periodically reduced by "reinversion," the reconstruction of a product-form inverse from A in a minimal sequence of pivots. The routines used here reinvert automatically at those points they determine will minimize the total operation count for the calculation.

In summary, the formulas of Table 8-1 indicate the dependence of the number of operations per iteration on problem size.

Table 8-1

Number of Operations per Iteration

Standard form	$\theta_1 (M + K) (N - M)$
Explicit form	$\theta_2 (M + K) (N - M) + \theta_3 (M + K)^2 + \theta_4 (M + K)^2$
Product form	$\theta_2 (M + K) (N - M) + \theta_5 (M + K)^2 + \theta_6 (M + K)^2$

It is beyond the scope of this study to discuss the factors θ of these formulas in detail. They will be used instead as guides to the scaling of our operation counts. Since for our problems N is closely proportional to M (N/M ranges from 1.67 to 3.43, averaging 2.31 with coefficient of variation 0.27), each of the

formulas has, approximately, $(M + K)^2$ as a common factor. Thus, comparative data for the three forms can be obtained as follows: for each problem, divide the total number of operations required to solve it by the number of iterations, and divide the result by $(M + K)^2$. The data of Table 8-2 were obtained in that way; for all forms, the ordinary simplex algorithm was used, and an S basis.

Table 8-2

Operations per Iteration / $(M + K)^2$

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B	avg.	c.v.
Standard form (run 12)	.88	3.00	2.11	.44	1.28	.79	.76	.59		1.23	.7
Explicit form (run 21)	.71	.95	.77	.49	.54	.31	.38	.29	.67	.57	.4
Product form (run 56)	.56	1.00	.54	.32	.45	.27	.25	.14	.26	.42	.4

The decrease of the ratios with size of problem is noteworthy; it is probably due to the decrease of the proportion of nonzero matrix entries. Table 8-3 makes a more direct comparison of these data, using the explicit form run as a base. Note that the relative efficiency of the product form tends to increase with the size of problem, owing, we think, to its greater ability to take advantage of the lower density of nonzeros.

Table 8-3

Standard and Product Compared with Explicit Form

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B	avg.	c.v.
Standard form	1.28	4.67	3.03	.83	2.30	2.57	2.04	2.63		2.42	.4
Product form	.82	1.05	.74	.64	.79	.86	.67	.45	.43	.71	.3

IX. ALGORITHMS COMPARED BY OPERATIONS

The algorithms of Sec. VI may finally be compared in the total number of operations they require to solve a problem. In Table 9-1 they are all compared with the ordinary algorithm in explicit form (run 21). With the exception of that procedure, each algorithm given has been run in that form of the simplex method best suited to it; the ratio pricing and the greatest-change (with F basis) procedures are omitted because they were not.

Table 9-1

Various Algorithms Compared with Ordinary, Explicit

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B	avg.	c.v.
						Singleton basis					
PN1--standard run 10	.75	2.14	2.42	.67	1.54	1.21	1.32	1.37		1.43	.4
PN2--standard run 14	1.07	2.53	2.84	.53	1.79	1.52	1.18	1.82		1.66	.4
Greatest-change standard run 15	1.15	3.66	3.72	.33	2.37	.99	1.66	2.36		2.03	.6
Ordinary--product run 56	.82	1.05	.74	.64	.79	.86	.67	.45	.43	.71	.3
						Full basis					
Symmetric standard run 40	.57	4.80	6.04	.47	2.81	2.64	1.01	1.42		2.47	.8
Ordinary--product run 55	.63	1.48	.81	.44	.89	.86	.25	.37	.38	.68	.5

We think that these figures constitute the best overall assessment of these alternative algorithms from the point of view of calculation needed. The product form of the ordinary algorithm seems definitely superior, with

use of a full basis probably being worthwhile for the larger problems.

We may try to predict the operation count for an unknown problem of given size. In Table 9-2, the counts of run 21 have been scaled in a manner intended to eliminate most of the influence of the size of the problem. Using the factor $(M + K)^2$ as in Table 8-2 to scale the count per iteration, and the factor M as in Table 6-3 to scale the number of iterations, we obtain the quotients of 9-2. The corresponding quotients for the other runs can be obtained by multiplying those of 9-1 by these numbers; the averages and coefficients of variation for those ratios are given in Table 9-3.

Table 9-2

Operations / $M(M + K)^2$ for Ordinary Explicit

Problem	1D	2A	1E	1A	5A	1G	1F	2B	1B	avg.	c.v.
	1.42	1.58	1.32	.62	.59	.40	.70	.34	2.22	1.02	.6

Table 9-3

Operations / $M(M + K)^2$ for Other Runs

Algorithm	Form	Basis	Run	Average	C.V.
PM1	standard	S	10	1.36	.8
PM2	"	S	14	1.59	.9
Greatest-change	"	S	15	2.04	1.0
Ordinary	product	S	56	.73	.6
Symmetric	standard	F	40	2.57	1.2
Ordinary	product	F	55	.73	.9

Since in practice K is usually 1, we can say that around M^3 operations are required to solve a linear programming problem. A rough minimum for problems of no more than some 100 constraints is 0.3 M^3, and 2 M^3 is a rough maximum for smaller problems. A count of more than 3 M^3 indicates an uncommonly hard problem or a rather poor algorithm.

X. CONCLUSION

Three kinds of data have been used above: iterations, operations, and passes. We have come to the view that iterations alone is the least informative: on the one hand, the operation count measures the total work of a routine, and on the other passes measure the amount of data handled. Of course, except for those of Sec. VII, sub-optimization is not used in any of the routines studied, so that in general the number of passes is equal to the number of iterations, which is the number we usually cite.

The results of Sec. IV show that use of a full basis will reduce the iterations taken in Phase One. (In Sec. IX, however, we found that it is of little value in reducing the operation count for the most efficient procedure.) It appears that there is no excuse for using an entirely artificial basis.

In Sec. V we failed to find any measure of infeasibility with which to conduct Phase One which works better than the ordinary measure--the sum of all the infeasibilities.

The results of Sec. VI show the positive-normalized procedures best in terms of iteration count, and that the full basis is good for the overall problem. The first conclusion is consistent with the interesting results of Kuhn and Quandt,[16] who have experimented with several pivot-column selection procedures on a large number of randomly-generated linear programming problems of special

type having up to 25 constraints. In the only place where their results can be matched with ours, we agree in ordering these procedures in increasing effectiveness in iteration count: ordinary, greatest-change, and positive-normalized. Our data suggest M and 3M as bounds for the number of iterations to solve a problem starting from a singleton basis.

The extent to which suboptimization will be of value in a routine depends considerably on how its data-handling is organized. Section VII shows that it can be used with little harm and under some circumstances with benefit to the total computational labor.

The comparison of operations per iteration in Sec. VIII shows pretty definitely that the order of the three main forms of the simplex method in increasing efficiency is: standard, explicit, product. The fact that those algorithms which are better than the ordinary in iterations need data which are conveniently obtained only in the standard form makes them less attractive from the point of view of operation count; Sec. IX shows that the ordinary algorithm in product form leads the rest. There are other considerations, however, for general uses of a linear programming routine, which are hard to evaluate properly but which argue for the standard form: in that form most of the data needed for the usual postoptimal analyses--reduced costs, etc.--are immediately available and need not be especially calculated.

An important fact about the product form, whose detailed study is beyond the scope of this report, is that the product-form inverse is extremely compact for problems of low density. This fact has considerable bearing on the choice of a routine for larger problems. SHARE problem 4A, having 245 constraints, can be solved with an all-in-core routine[9] for the IBM 7090, which has 32,768 words of core. A similar routine using the explicit form would require 75,000 words, and using the standard form, 118,000 words.

At this time, we feel that a product-form routine employing the ordinary or the greatest-change algorithm with sub optimization, with option for using a full basis, will pull together the best features of the procedures we have studied so far.

It may seem disappointing that our results have not allowed a more decisive ordering of the proposals studied. In part, of course, this is due to our having selected the more promising possibilities from a larger number of candidates; but it may also be the case that, as linear programming is presently understood, it is not possible to do a great deal better than some of these procedures do. A linear programming method has two parts: find the optimal basis, and calculate the optimal solution. If the optimal basis were known, it would still in the general case require some $\frac{1}{3}M^3$ operations to solve the linear equations thus identified

(although a product-form method would do much better for problems, like ours, having a low density of data). Since some of our procedures do the whole job in about M^3 operations, there does not seem to be an enormous amount of room for improvement.

Appendix

THE SCRMP RUNS AND DATA

NATURE OF RUN

(Abbreviations used: for bases--None N, Singleton S, Full F; for forms of the simplex method--Standard S, Explicit E, Product P.)

Run	Starting basis	Form	
5	N	P	Ordinary
6	S	E	Ratio pricing
8	N	E	"
10	S	S	Positive-normalized 1
12	S	S	Ordinary
14	S	S	Positive-normalized 2
15	S	S	Greatest-change
21	S	E	Ordinary
22	S	S	Ordinary with suboptimization; L = 2
27	S	S	" 3
28	S	S	" 5
29	S	S	" 8
23	S	S	Greatest-change with suboptimization; L = 2
24	S	S	" 3
25	S	S	" 5
26	S	S	" 8
31	S	E	Sequential Phase One
32	F	E	Fudge Phase One
33	S	E	Least-infeasibility Phase One
36	F	E	Sequential Phase One
37	F	E	Least-infeasibility Phase One
38	F	E	Extended composite Phase One
39	F	E	Ordinary
40	F	S	Symmetric
41	F	E	Greatest-change
42	S	S	PM1 with suboptimization; L = 2
43	S	S	" 3
55	F	P	Ordinary
56	S	P	Ordinary
59	F	S	Parametric

DATA

(These abbreviations are used: p1, iterations in Phase One; p2, total iteration count to solve problem; pa, number of passes to solve problem; op, total operation count to solve problem, where "K" stands for "000".)

Run	Problem Data	1A	1B	1D	1E	1F	1G	2A	2B	5A
5	p2	70	443	61	66	184		114	221	
6	p1	33		32	75	66	37	1	125	0
	p2	34		32	75	66	51	81	152	35
8	p1	40	306	37	88	113	92	69	151	46
	p2	40	365	37	88	113	92	99	151	49
10	p1	25		24	27	42	17	1	62	0
	p2	42		41	51	102	41	40	93	27
	op	15956		22532	151K	272K	55378	104K	440K	37692
12	p2	39		56	59	124	62	74	143	36
	op	19750		38456	189K	421K	118K	227K	845K	56327
14	p2	39		45	52	80	45	41	102	28
	op	12623		32297	177K	243K	6.9764	123K	585K	43829

Run	Problem Data	1A	1B	1D	1E	1F	10	2A	2B	5A
15	p1	24		27	43	50	18	1	97	0
	p2	32		38	65	88	40	55	137	32
	op	7717		34449	232K	342K	45491	178K	758K	58127
21	p1	25	253	25	34	55	21	0	77	0
	p2	42	390	54	53	121	62	50	113	37
	op	23682	3620K	30045	62374	206K	45908	48569	321K	24502
22	p2	39		63	56	126	63	81	119	48
	pa	24		35	36	79	40	48	73	34
23	p2	38		56	64	111	53	61	108	53
	pa	25		35	43	82	36	42	73	37
24	p2	36		52	63	125	61	56	103	58
	pa	18		28	40	69	36	33	58	26
25	p2	33		50	75	104	57	57	102	63
	pa	12		22	36	43	26	24	37	25
26	p2	31		56	82	127	54	57	105	64
	pa	10		20	33	43	19	21	30	22
27	p2	39		64	71	141	73	98	145	43
	pa	16		28	37	68	35	45	64	22
28	p2	39		69	70	123	79	79	140	52
	pa	15		23	25	39	27	30	44	21
29	p2	37		56	97	143	71	98	131	48
	pa	9		14	31	38	17	31	38	14
31	p1	38	253	24	32	54	21		75	
32	p1	15	135	10	20	6	34		71	
33	p1	34	238	27	35	68	22		86	
36	p1	14	145	9	32	6	51		80	
37	p1	12	186	9	33	6	50		80	
38	p1	17	136	8	26	3	22		55	
39	p1	12	134	8	24	3	34	0	53	0
	p2	22	236	34	49	54	75	62	86	37
	op	12439	3245K	14015	79322	79285	113K	70660	239K	24502
40	p2	19		23	81	59	46	41	83	31
	op	11193		17012	377K	208K	121K	233K	455K	68736
41	p1	7	80	11	18	3	22	0	88	0
	p2	23	105	21	36	46	58	41	102	31
42	p2	42		58	68	146	66	67	108	48
	pa	24		33	43	94	39	43	67	34
43	p2	39		62	76	141	70	77	123	49
	pa	16		28	39	67	34	36	55	26
55	p2	21	285	38	35	54	59	66	86	38
	op	10342	1387K	18913	50954	52300	39554	72038	119K	21883
56	p2	41	421	56	56	121	62	50	106	35
	op	15050	1540K	24531	45923	137K	39511	51179	144K	19278
59	p2	24		21	80	77	115	87	108	38

REFERENCES

1. Dantzig, George B., "Maximization of a Linear Function of Variables Subject to Linear Inequalities," Activity Analysis of Production and Allocation, T. C. Koopmans, ed., Wiley, 1951, pp. 339-347.
2. Wolfe, Philip, RSM1 Linear Programming Routine, SHARE Distribution Agency, #863, March 1960.
3. Wolfe, P., Status Report: The SCEMP Project, The RAND Corporation, March 1961.
4. Smith, D. M., Results on SHARE Test Problems, Unpublished.
5. Gass, S. I., Linear Programming--Methods and Applications, McGraw-Hill, 1958.
6. Hadley, George, Linear Programming, Addison-Wesley, 1962.
7. Orchard-Hays, William, A Composite Simplex Algorithm--II, The RAND Corporation, RM-1275, May 1954.
8. Wolfe, P., A Technique for Resolving Degeneracy in Linear Programming, The RAND Corporation, RM-2995, May 1962.
9. Clasen, Richard J., RSMFOR Linear Programming Routine, SHARE Distribution Agency, 1962.
10. Wolfe, P., An Extended Composite Algorithm for Linear Programming, The RAND Corporation, P-2373, July 1961.
11. Dickson, J. C., and F. P. Frederick, "A Decision Rule for Improved Efficiency in Solving Linear Programming Problems with the Simplex Algorithm," Comm. Assoc. Comp. Mach., Vol. 3, September 1960, pp. 509-512.
12. Talacko, J. V., and R. T. Rockafellar, A Compact "Simplex" Algorithm for General Linear Programs, August 1960, Unpublished.
13. Dantzig, George B., Linear Programming and Extensions, Princeton University, 1963.
14. Orchard-Hays, W., and D. M. Smith, "Computational Efficiency in Product Form Linear Programming Codes," Proceedings of the 1962 Symposium on Mathematical Programming, R. L. Graves and P. Wolfe, eds., McGraw-Hill, (to be published).
15. Harvey, Roy, et al., SCM3 Linear Programming Routine, SHARE Distribution Agency, 1962.
16. Kuhn, H. W., and R. Quandt, An Experimental Study of the Simplex Method, Unpublished.

LIST OF RAND NOTES ON LINEAR PROGRAMMING AND EXTENSIONS

RM-1264 Part 1: The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Restraints, by G. B. Dantzig, A. Orden, and P. Wolfe, April 5, 1954. Published in the Pacific Journal of Mathematics, Vol. 5, No. 2, June, 1955, pp. 183-195. (ASTIA No. AD 114134)

RM-1265 Part 2: Duality Theorems, by G. B. Dantzig and A. Orden, October 30, 1953. (ASTIA No. AD 114135)

RM-1266 Part 3: Computational Algorithm of the Simplex Method by G. B. Dantzig, October 26, 1953. (ASTIA No. AD 114136)

RM-1267-1 Part 4: Constructive Proof of the Min-Max Theorem, by G. B. Dantzig, September 8, 1954. Published in the Pacific Journal of Mathematics, Vol. 6, No. 1, Spring, 1956, pp. 25-33. (ASTIA No. AD 114137)

RM-1268 Part 5: Alternate Algorithm for the Revised Simplex Method Using Product Form for the Inverse, by G. B. Dantzig and W. Orchard-Hays, November 19, 1953. (ASTIA No. AD 90500)

RM-1440 Part 6: The RAND Code for the Simplex Method (SX4) (For the IBM 701 Electronic Computer), by William Orchard-Hays, February 7, 1955. (ASTIA No. AD 86718)

RM-1270 Part 7: The Dual Simplex Algorithm, by G. B. Dantzig, July 3, 1954. (ASTIA No. AD 114139)

RM-1367 Parts 8, 9, and 10: Upper Bounds, Secondary Constraints, and Block Triangularity in Linear Programming, by G. B. Dantzig, October 4, 1954. Published in Econometrica, Vol. 23, No. 2, April, 1955, pp. 174-183. (ASTIA No. AD 111054)

RM-1274 Part 11: Composite Simplex-Dual Simplex Algorithm--I, by G. B. Dantzig, April 26, 1954. (ASTIA No. AD 114140)

RM-1275 Part 12: A Composite Simplex Algorithm--II, by William Orchard-Hays, May 7, 1954. (ASTIA No. AD 114141)

RM-1281 Part 13: Optimal Solution of a Dynamic Leontief Model with Substitution, by G. B. Dantzig, June 15, 1954. Published in Econometrica, Vol. 23, No. 3, July, 1955, pp. 295-302. (ASTIA No. AD 90501)

RM-1290 Part 14: A Computational Procedure for a Scheduling Problem of Edie, by G. B. Dantzig, July 1, 1954. Published in Operations Research, Vol. 2, No. 3, August, 1954, pp. 339-341. (ASTIA No. AD 109960)

RM-1328 Part 15: Minimizing the Number of Carriers to Meet a Fixed Schedule, by G. B. Dantzig and D. R. Fulkerson, August 24, 1954. Published in Naval Research Logistics Quarterly, Vol. 1, No. 3, September, 1954, pp. 217-222. (ASTIA No. AD 109960)

RM-1369 Part 16: The Problem of Routing Aircraft--A Mathematical Solution, by A. R. Ferguson and G. B. Dantzig, September 1, 1954. Published in Aeronautical Engineering Review, Vol. 14, No. 4, April, 1955, pp. 51-55. (ASTIA No. AD 90504)

RM-1374 Part 17: Linear Programming under Uncertainty, by G. B. Dantzig, November 16, 1954. Published in Management Science, Vol. 1, Nos. 3-4, April-July, 1955, pp. 197-206. (ASTIA No. AD 90495)

RM-1375 Part 18: Status of Solution of Large-scale Linear Programming Problems, by G. B. Dantzig, November 30, 1954. (ASTIA No. AD 86396)

RM-1383 Part 19: The Fixed-Charge Problem, by W. M. Hirsch and G. B. Dantzig, December 1, 1954. (ASTIA No. AD 90494)

RM-1400 Part 20: Maximal Flow through a Network, by L. R. Ford and D. R. Fulkerson, November 19, 1954. Published in Canadian Journal of Mathematics, Vol. 8, No. 3, 1956, pp. 399-404. (ASTIA No. AD 90541)

RM-1418-1 Part 21: On the Min Cut Max Flow Theorem of Networks, by G. B. Dantzig and D. R. Fulkerson, April 15, 1955. Published in Linear Inequalities and Related Systems, Annals of Mathematics Study No. 38, edited by H. W. Kuhn and A. W. Tucker, Princeton University Press, 1956, pp. 215-221. (ASTIA No. AD 86705)

RM-1475 Part 22: Recent Advances in Linear Programming, by G. B. Dantzig, April 12, 1955. Published in Management Science, Vol. 2, No. 2, January, 1956, pp. 131-144. (ASTIA No. AD 111056)

RM-1432 Part 23: A Production Smoothing Problem, by S. M. Johnson and G. B. Dantzig, January 6, 1955. Published in Proceedings of the Second Symposium in Linear Programming (Washington, D. C., January 27-29, 1955), Vol. 1, U. S. Department of Commerce, Washington, D. C. 1956, pp. 151-176. (ASTIA No. AD 90506)

RM-1470 Part 24: The Modification of the Right-hand side of a Linear Programming Problem, by H. M. Markowitz, April 20, 1955. (ASTIA No. AD 90543)

RM-1452 Part 25: The Elimination Form of the Inverse and Its Application to Linear Programming, by H. M. Markowitz, April 8, 1955. (ASTIA No. AD 86956)

RM-1489 Part 26: Computation of Maximal Flows in Networks, by D. R. Fulkerson and G. B. Dantzig, April 1, 1955. Published in Naval Research Logistics Quarterly, Vol. 2, No. 4, December, 1955, pp. 277-283. (ASTIA No. AD 90548)

RM-1553 Part 27: Dilworth's Theorem on Partially Ordered Sets, by A. J. Hoffman and G. B. Dantzig, August 26, 1955. Published in Linear Inequalities and Related Systems, Annals of Mathematics Study No. 38, edited by H. W. Kuhn and A. W. Tucker, Princeton University Press, 1956, pp. 207-214. (ASTIA No. AD 88670)

RM-1560 Part 28: A Simple Linear Programming Problem Explicitly Solvable in Integers, by O. A. Gross, September 30, 1955. (ASTIA No AD 90546)

RM-1604 Part 29: A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem, by L. R. Ford and D. R. Fulkerson, December 29, 1955. Published in Canadian Journal of Mathematics, Vol. 9, 1957, pp. 210-218. (ASTIA No. AD 90545)

RM-1644 Part 30: A Class of Discrete-type Minimization Problems, by O. A. Gross, February 24, 1956. (ASTIA No. AD 90560)

RM-1709 Part 31: A Primal-Dual Algorithm, by G. B. Dantzig, L. R. Ford, and D. R. Fulkerson, May 9, 1956. Published in Linear Inequalities and Related Systems, Annals of Mathematics Study No. 38, edited by H. W. Kuhn and A. W. Tucker, Princeton University Press, 1956, pp. 171-181. (ASTIA No. AD 111635)

RM-1736 Part 32: Solving the Transportation Problem, by L. R. Ford and D. R. Fulkerson, June 20, 1956. Published in Management Science, Vol. 3, No. 1, October, 1956, pp. 24-32. (ASTIA No. AD 111816)

RM-1737 Part 33: A Theorem on Flows in Networks, by David Gale, June 22, 1956. Published in Pacific Journal of Mathematics, Vol. 7, No. 2, 1957, pp. 1073-1082. (ASTIA No. AD 112371)

RM-1798 Part 34: A Primal-Dual Algorithm for the Capacitated Hitchcock Problem, by L. R. Ford and D. R. Fulkerson, September 25, 1956. Published in Naval Research Logistics Quarterly, Vol. 4, No. 1, March, 1957, pp. 47-54. (ASTIA No. AD 112373)

RM-1832 Part 35: Discrete-variable Extremum Problems, by G. B. Dantzig, December 6, 1956. Published in Operations Research, April, 1957. (ASTIA No. AD 112411)

RM-1833 Part 36: The Allocation of Aircraft to Routes--An Example of Linear Programming under Uncertain Demand, by A. R. Ferguson and G. B. Dantzig, December 7, 1956. (ASTIA No. AD 112418)

RM-1799 Part 37: Concerning Multicommodity Networks, by J. T. Robacker, September 26, 1956. (ASTIA No. AD 112392)

RM-1864 Part 38: Note on B. Klein's "Direct Use of Extremal Principles in Solving Certain Problems Involving Inequalities," by G. B. Dantzig, January 29, 1957. Published in Operations Research, April, 1956. (ASTIA No. AD 123515)

RM-1859 Part 39: Slightly Intertwined Linear Programming Matrices, by Richard Bellman, January 23, 1957. Published in Management Science, July, 1957. (ASTIA No. AD 123533)

RM-1977 Part 40: Network Flows and Systems of Representatives, by L. R. Ford and D. R. Fulkerson, September 12, 1957. Published in Canadian Journal of Mathematics, Vol. 10, No. 1, 1958, pp. 78-84. (ASTIA No. AD 144263)

RM-1981 Part 41: Constructing Maximal Dynamic Flows from Static Flows, by L. R. Ford and D. R. Fulkerson, September 17, 1957. Published in Operations Research, Vol. 6, No. 3, May-June, 1958, pp. 419-433. (ASTIA No. AD 144279)

RM-2021 Part 42: Linear Programming and Structural Design, by W. Prager, December 3, 1957. (ASTIA No. AD 150661)

RM-1976 Part 43: A Feasibility Algorithm for One-way Substitution in Process Analysis, by K. J. Arrow and S. M. Johnson, September 12, 1957. Published in Studies in Linear and Non-Linear Programming, by Kenneth J. Arrow, Leonid Hurwicz, and Hirofumi Uzawa, Stanford University Press, 1958, pp. 198-202. (ASTIA No. AD 144278)

RM-2152 Part 44: Transient Flows in Networks, by D. Gale, April 11, 1958. (ASTIA No. AD 150686)

RM-2159 Part 45: A Network-Flow Feasibility Theorem and Combinatorial Applications, by D. R. Fulkerson, April 21, 1958. Published in Canadian Journal of Mathematics, Vol. XI, No. 3, 1959. (ASTIA No. AD 156011)

RM-2178 Part 46: Bounds on the Primal-Dual Computation for Transportation Problems, by D. R. Fulkerson, May 21, 1958. (ASTIA No. AD 156001)

RM-2209 Part 47: Solving Linear Programs in Integers, by G. B. Dantzig, July 11, 1958. Published in Naval Research Logistics Quarterly, Vol. 6, No. 1, March, 1959. (ASTIA No. AD 156047)

RM-2287 Part 48: Inequalities for Stochastic Linear Programming Problems, by Albert Madansky, November 13, 1958. Published in Management Science, January, 1960. (ASTIA No. AD 208311)

RM-2321 Part 49: On a Linear Programming-Combinatorial Approach to the Traveling Salesman Problem, by G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, January 26, 1959. Published in Operations Research, Vol. 7, No. 1, January-February, 1959. (ASTIA No. AD 211642)

RM-2338 Part 50: On Network Flow Functions, by L. S. Shapley, March 16, 1959. (ASTIA No. AD 214635)

RM-2388 Part 51: The Simplex Method for Quadratic Programming, by Philip Wolfe, June 5, 1959. Published in Econometrica, Vol. 27, No. 3, July, 1959, pp. 382-398. (ASTIA No. AD 225224)

RM-2425 Part 52: Computing Tetraethyl-Lead Requirements in the Linear-Programming Format, by G. B. Dantzig, T. T. Kawaratani, and R. J. Ullman, April 1, 1960. Published in Operations Research, Vol. 8, No. 1, January-February, 1960. (ASTIA No. AD 237380)

RM-2480 Part 53: On the Equivalence of the Capacity-Constrained Transshipment Problem and the Hitchcock Problem, by D. R. Fulkerson, January 13, 1960. (ASTIA No. AD 235811)

RM-2597 Part 54: An Algorithm for the Mixed Integer Problem. by Ralph Gomory, July 7, 1960. (ASTIA No. AD 243212)

RM-2751 Part 55: On the Solution of Two-Stage Linear Programs under Uncertainty, by George Dantzig and Albert Madansky, July 1961. (ASTIA No. AD 263219)

RM-2752 Part 56: Methods of Solution of Linear Programs under Uncertainty, by Albert Madansky, April 1961 (ASTIA No. 257816)

RM-2813 Part 57: The Decomposition Algorithm for Linear Programming, by George Dantzig and Philip Wolfe, August 1961. (ASTIA No. AD 263628)

RM-2956 Part 58: An Algorithm for Scaling Matrices, by D. R. Fulkerson and Philip Wolfe, February 1962. (ASTIA No. AD 279143)

RM-2957 Part 59: Linear Programming in a Markov Chain, by G. B. Dantzig and Philip Wolfe, April 1962. (ASTIA No. AD 274595)

RM-2993 Part 60: A Linear Program of Prager's, by Oliver Gross, April 1962. (ASTIA No. AD 274596)

RM-2995 Part 61: A Technique for Solving Degeneracy in Linear Programming, by Philip Wolfe, May 1962. (ASTIA No. AD 275527)

RM-3199 Part 62: Simplex Method and Theory, by A. W. Tucker, June 1962. (ASTIA No. AD 277519)